systemd

Timothée Ravier
siosm@floss.social - tim.siosm.fr/cours - github.com/travier

5e année cycle ingénieur, filière STI
Option Sécurité des Systèmes Embarqués et du Cloud (2SEC)

2024 - 2025

Sommaire

Histoire : SystemV et sysinit (1983)

- Runlevels (`/etc/inittab`)
- Script shell pour chaque service (`/etc/init.d/nginx`), très souvent spécifique à chaque distribution
- Démarrage séquentiel, arrêt séquentiel
- Difficile de maintenir des modifications dans la durée
- Aucune information sur l'état du système et les services démarrés
- Difficile de suivre les processus dans la durée (`/var/run/nginx.pid`)

Histoire : launchd (2005, Mac OS X 10.4)

- Démarre le système et gère les services
- Démarrage des services en parallèle et à la demande
- Démon launchd (PID 1) et « télécommande » `launchctl` pour gérer les services
- Gestion des daemons par utilisateur possible
- Configuration en XML

Histoire : Upstart (2006, Ubuntu 6.10)

- Démarrage des service à la suite d'événements
- La carte bluetooth est disponible $\Rightarrow$ démarre le service bluetooth
- Le réseau est disponible $\Rightarrow$ montage des disques distants
- Difficile d'empêcher un service de démarrer automatiquement
- Meilleur que SysV, mais pas assez puissant

Histoire : et les autres…

Implémentations alternatives, généralement basées sur un langage de script :

- OpenRC (2007, Gentoo)
- rcNG (NetBSD, FreeBSD et DragonFlyBSD)
- runit
- Service Management Facility (SMF, OpenIndiana)

https://linuxfr.org/news/petit-etat-de-l-art-des-systemes-d-initialisation-1

systemd

- Première version annoncée le 30 mars 2010 par Lennart Poettering
- Adopté comme `init` par défaut par pratiquement toutes les distributions majeures (RHEL, Fedora, Ubuntu, Debian, openSUSE, Arch Linux, NixOS, etc.)
- Optionnel pour Gentoo (openrc par défaut)
- Non disponible sur Alpine, Slackware, Void

systemd

- Au début : juste un remplaçant d'`init`
- Désormais : ensemble de blocs de base pour construire un système Linux
- Adapté à tous les environnements : Distributions embarquées, voitures, téléphones, tablettes, supercalculateurs, desktops, etc.
- Beaucoup de rumeurs et fausses informations circulent à propos de systemd : The Biggest Myths about systemd

Fonctionnement

Fonctionnalités

- Gestionnaire des **services** et du **système** : vue globale
- Contrôle des **« units »** plutôt que des démons ou processus
- Gestion des dépendances entre les units
- Suivi des processus qui composent chaque unit :
  - Correspondance unit ⇔ processus
  - Arrêt **maîtrisé** des units
- Temps de démarrage minimal
- Debuggable : Reproductibilité, récupération de tous les logs du système
- Interface utilisateur : `systemctl`

Init : PID 1

- « Premier » processus lancé par le noyau
- Succède dans le cas général à l'*initrd*
- Chargé de démarrer le reste du système :
  - montage des partitions supplémentaires (`/` et `/usr` montées par l'*initrd*)
  - démarrage des services systèmes
- Parent de tous les processus orphelins

Init : systemd PID 1

- Spécialiste du **lancement de processus** :
  - **Quoi?** service nginx, mysql, etc.
  - **Comment?** UID, GID, CWD, capabilities, env, etc.
  - **Quand?** au démarrage, gestion des dépendances, activation suite à un timer, une socket, un message D-Bus, etc.
  - **Qui?** Logs
- et de leur **suivi** :
  - **Qui?** Quels processus font parti de mon service ?
  - **Status :** watchdogs, socket de notification
  - **Redémarrage :** en cas d'échec, à la réception d'un SIGSEGV
  - **Arrêt propre :** aucun processus oublié

cgroups

- systemd place les **processus** de chaque **unit** dans un **cgroup distinct**
- systemd « seul » gestionnaire des cgroups
- Changement de granularité : processus $\Rightarrow$ service (cgroup)
- Réponse facile aux questions :
  - A quel service appartient ce processus ?
  - Quels processus font partie de ce service ?

cgroups: `systemd-cgls`

```
Control group /:
-.slice
├─init.scope
│ └─1 /usr/lib/systemd/systemd --switched-root --system --deserialize 24
├─machine.slice
│ └─machine-qemu\x2dfedora22.scope
│   └─11026 /usr/sbin/qemu-system-x86_64 -name fedora22 -S -machine pc-i440fx-2.3,accel=kvm...
├─system.slice
│ ├─dbus.service
│ │ └─569 /usr/bin/dbus-daemon --system --address=systemd: --nofork --nopidfile --systemd-activation
│ ├─systemd-journald.service
│ │ └─309 /usr/lib/systemd/systemd-journald
│ ├─systemd-udevd.service
│ │ └─339 /usr/lib/systemd/systemd-udevd
└─user.slice
  └─user-1000.slice
    ├─user@1000.service
    │ ├─pulseaudio.service
    │ │ ├─1446 /usr/bin/pulseaudio --daemonize=no
    │ │ └─1466 /usr/lib/pulse/gconf-helper
    │ ├─gpg-agent.service
    │ │ └─1304 /usr/bin/gpg-agent --daemon --enable-ssh-support --pinentry-program /usr/bin/pinentry-qt
    │ ├─init.scope
    │ │ ├─1289 /usr/lib/systemd/systemd --user
    │ │ └─1290 (sd-pam)
    └─session-c2.scope
      ├─ 1411 kwin_x11
      ├─ 6255 /usr/bin/tmux
      ├─19664 /usr/bin/dolphin
      ├─24500 /usr/bin/thunderbird
      ├─27086 /usr/bin/zsh
```

11 types d'units systemd

- nginx.service
- multi-user.target
- sshd.socket
- systemd-tmpfiles-clean.timer
- -.mount, home.mount
- acpid.path
- system.slice, user-1000.slice
- session-1.scope
- sys-devices-virtual-block-dm\x2d0.device
- proc-sys-fs-binfmt_misc.automount
- dev-mapper-system\x2dswap.swap

Units : Emplacements

- `(/usr)/lib/systemd/system` : Fournies par la distribution (lecture seule);
- `/run/systemd/system` : Générées à l'exécution, non persistantes;
- `/etc/systemd/system` : Crées et modifiées par l'administrateur.
- Ordre de priorité : `/etc/` $\rightarrow$ `/run/` $\rightarrow$ `/usr/lib/`
- Liste des units :
  - installées : `systemctl list-unit-files`
  - chargées : `systemctl list-units`

Units : httpd.service

```
# systemctl cat httpd.service
```

```
[Unit]
Description=Apache Web Server
After=network.target remote-fs.target nss-lookup.target

[Service]
Type=forking
PIDFile=/run/httpd/httpd.pid

ExecStart=/usr/bin/apachectl start
ExecStop=/usr/bin/apachectl graceful-stop
ExecReload=/usr/bin/apachectl graceful

PrivateTmp=true
LimitNOFILE=infinity

[Install]
WantedBy=multi-user.target
```

Units : httpd.service

```
# systemctl show -p Description httpd.service
```

```
[Unit]
Description=Apache Web Server
```

Units : httpd.service

```
# systemctl start httpd.service
```

```
[Unit]
Description=Apache Web Server

[Service]
Type=forking
PIDFile=/run/httpd/httpd.pid

ExecStart=/usr/bin/apachectl start
```

Units : httpd.service

```
# systemctl stop httpd.service
```

```
[Unit]
Description=Apache Web Server

[Service]
Type=forking
PIDFile=/run/httpd/httpd.pid

ExecStart=/usr/bin/apachectl start
ExecStop=/usr/bin/apachectl graceful-stop
```

```
# systemctl kill --signal=SIGTERM httpd.service
```

Units : httpd.service

```
# systemctl reload httpd.service
```

```
[Unit]
Description=Apache Web Server

[Service]
Type=forking
PIDFile=/run/httpd/httpd.pid

ExecStart=/usr/bin/apachectl start
ExecStop=/usr/bin/apachectl graceful-stop
ExecReload=/usr/bin/apachectl graceful
```

systemctl status

```
$ sudo systemctl status cronie.service
● cronie.service - Periodic Command Scheduler
   Loaded: loaded (/usr/lib/systemd/system/cronie.service; enabled; vendor preset: disabled)
   Active: active (running) since mar. 2016-01-05 03:01:07 CET; 1 weeks 1 days ago
 Main PID: 575 (crond)
    Tasks: 1 (limit: 512)
   CGroup: /system.slice/cronie.service
           └─575 /usr/bin/crond -n

janv. 12 17:48:01 hydra anacron[23364]: Job `cron.weekly' terminated
janv. 12 18:01:01 hydra CROND[23982]: (root) CMD (run-parts /etc/cron.hourly)
janv. 12 19:01:01 hydra CROND[26767]: (root) CMD (run-parts /etc/cron.hourly)
janv. 13 13:01:01 hydra CROND[14366]: (root) CMD (run-parts /etc/cron.hourly)
janv. 13 13:01:01 hydra anacron[14371]: Anacron started on 2016-01-13
janv. 13 13:01:01 hydra anacron[14371]: Will run job `cron.daily' in 10 min.
janv. 13 13:01:01 hydra anacron[14371]: Jobs will be executed sequentially
janv. 13 13:40:53 hydra anacron[14371]: Job `cron.daily' started
janv. 13 14:01:01 hydra CROND[15078]: (root) CMD (run-parts /etc/cron.hourly)
janv. 13 15:01:01 hydra CROND[8012]: (root) CMD (run-parts /etc/cron.hourly)
```

Gestion des dépendances

- Dépendance : **Requires=** / **Wants=**
  - `systemctl show -p Requires nginx`
  - `systemctl show -p Wants nginx`
  - `systemctl list-dependencies nginx`
  - `systemctl list-dependencies --reverse nginx`
- Ordre : **After=** / **Before=** :
  - `systemctl show -p After nginx`
  - `systemctl show -p Before nginx`
  - `systemctl list-dependencies --after nginx`
  - `systemctl list-dependencies --before nginx`

Comment ordonner le lancement d'un service?

Démarrer nginx après mysql et seulement si son lancement a réussi :

```
$ cat /etc/systemd/system/nginx.service.d/after-mysql.conf
```

```
[Unit]
Requires=mysql.service
After=mysql.service
```

Units : target

- Remplaçant des runlevels SysVinit
- Points de synchronisation
- Regroupe plusieurs units à l'aide des dépendances
- Plus flexible (pas de limite en quantité et simultanéité)

Units : httpd.service

```
[Unit]
Description=Apache Web Server
After=network.target remote-fs.target nss-lookup.target

[Service]
Type=forking
PIDFile=/run/httpd/httpd.pid

ExecStart=/usr/bin/apachectl start
ExecStop=/usr/bin/apachectl graceful-stop
ExecReload=/usr/bin/apachectl graceful

[Install]
WantedBy=multi-user.target
```

Usage : Démarrage au boot

```
# systemctl enable httpd.service
ln -s /usr/lib/systemd/system/httpd.service \
    /etc/systemd/system/multi-user.target.wants/
```

```
# systemctl disable httpd.service
rm /etc/.../multi-user.target.wants/httpd.service
```

```
# systemctl mask httpd.service
ln -s /dev/null /etc/systemd/system/httpd.service
```

Pages de manuel correspondantes

- Toutes les pages de man sont listées dans systemd.index(7)
- Toutes les options d'unit sont listées dans systemd.directives(7)
- Options génériques pour toutes les units dans systemd.unit(5)
- Options spécifiques aux services dans systemd.service(5)
- Options spécifiques à l'exécution de processus dans systemd.exec(5)

journald

Affichage des logs du journal

- `journalctl -e` : affiche les logs les plus récents du boot courant
- `journalctl -b -1` : affiche les logs du boot précédent
- `journalctl -b -0 _PID=333` : limite l'affichage aux logs émis par le PID 333 pour ce démarrage
- `journalctl -k` : équivalent de `dmesg` (logs du noyau)
- `journalctl -e -u nginx.service` : affiche tous les logs liés au service nginx
- `journalctl -f -n 50` : affiche les 50 dernières lignes de log puis affiche les logs au fur et à mesure
- `journalctl /usr/bin/dbus-daemon` : tous les logs émis par le binaire `/usr/bin/dbus-daemon`

Résoudre les problèmes sur un système

Résoudre un problème avec un service ?

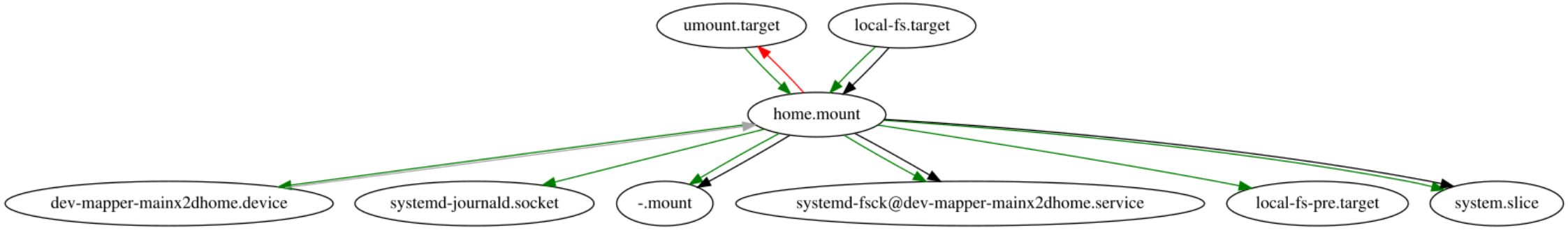
- `# systemctl --state failed`
- `# systemctl status foo.service`
- `# systemctl show foo.service`
- `# systemctl cat foo.service`

Résoudre un problème au démarrage ?

- Options noyau :
  - `systemd.unit=rescue.target`
  - `systemd.unit=emergency.target`
  - `systemd.log_level=debug systemd.log_target=kmsg log_buf_len=1M`
  - `systemd.debug-shell` (`tty9`)
  - `systemd.confirm_spawn=true`
- `# systemctl list-jobs`

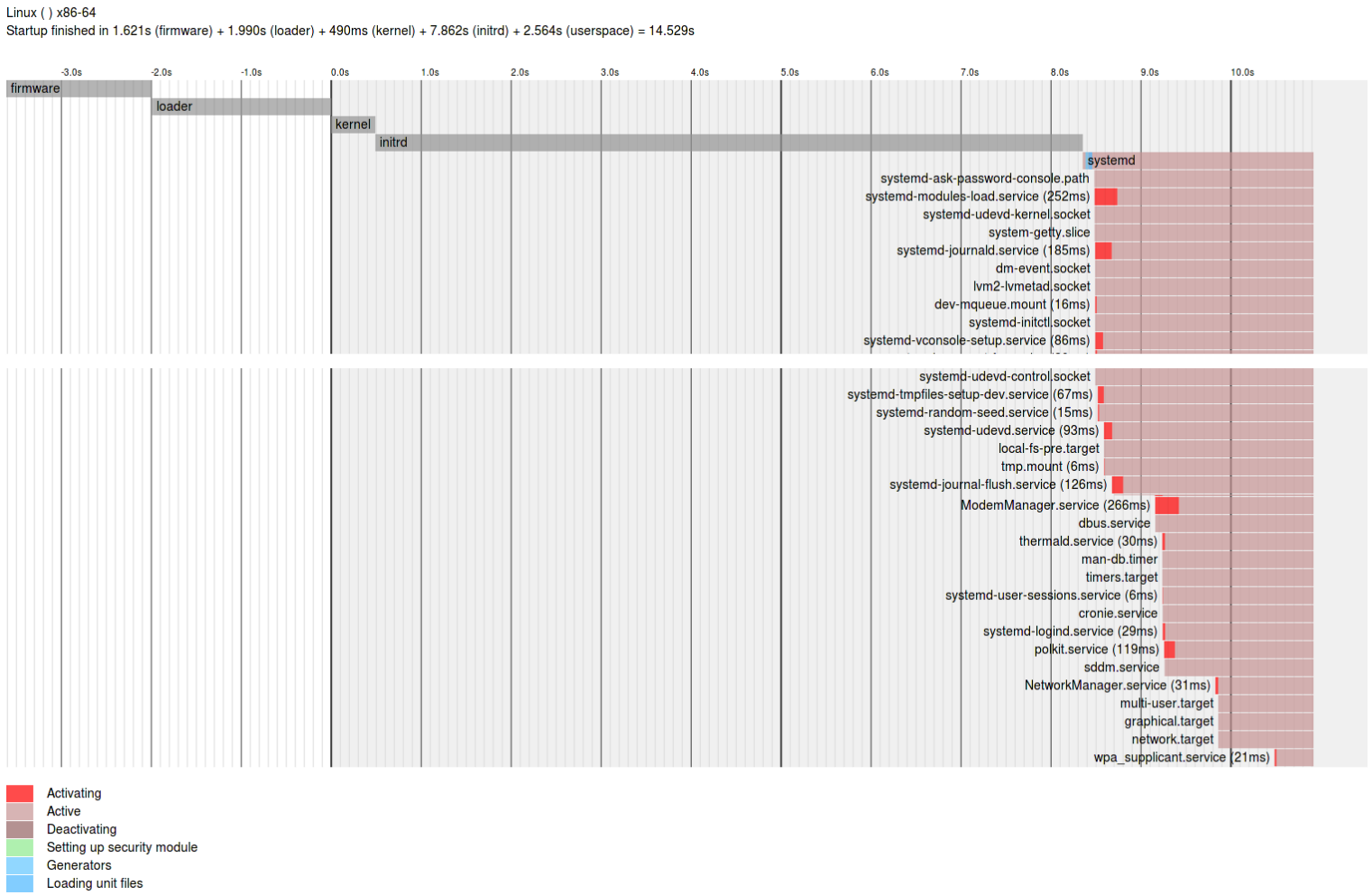
Problème de dépendance ?

```
# systemctl list-dependencies
# systemd-analyze dot home.mount | dot -Tsvg > plt.svg
$ firefox plt.svg
```



Problème de performance ?

```
# systemd-analyze plot > boot.svg
```



Problème de performance ?

```
$ systemd-analyze critical-chain
The time after the unit is active or started is printed after the "@" character.
The time the unit takes to start is printed after the "+" character.

graphical.target @1.505s
└─multi-user.target @1.505s
  └─NetworkManager.service @1.473s +31ms
    └─firewalld.service @800ms +672ms
      └─basic.target @798ms
        └─sockets.target @798ms
          └─sshd.socket @798ms
            └─sysinit.target @797ms
              └─systemd-timesyncd.service @763ms +33ms
                └─systemd-tmpfiles-setup.service @746ms +13ms
                  └─local-fs.target @745ms
                    └─boot.mount @731ms +13ms
                      └─systemd-fsck@dev-sda1.service @681ms +49ms
                        └─dev-sda1.device @680ms
```

Problème lors d'un précédent démarrage ?

Vérifier `/etc/systemd/journald.conf` :

```
[Journal]
...
Storage=persistent
```

```
# journalctl -b -1
```

logind, resolved, networkd, timesyncd, etc.

Autres projets rattachés

Essentiel :

- logind : Gère les sessions utilisateurs
- udevd : Gère les périphériques

Optionnel :

- resolved : Résolveur DNS
- networkd : Gestionnaire de la configuration réseau
- timesyncd : Client NTP (client uniquement)
- oomd : Réagir avant que le noyau manque de mémoire (*OOM*)
- homed : Gestion des utilisteurs intéractifs et de leur dossiers *home*

Sécurité

Fonctionnalité de sécurité

Support de « toutes » les options proposées par le noyau :

- UID, GID
- limites & Co
- cgroups v1 & v2
- namespaces
- capabilities
- seccomp-bpf
- NoNewPrivs
- etc.

Exemples : Durcissement système à l'aide de systemd (SSTIC 2017)

Sécurité de systemd ?

- Quelques vulnérabilités majeures : CVE-2018-15686, CVE-2018-15688, CVE-2016-7795, CVE-2016-10156, CVE-2013-4392, CVE-2013-4327, CVE-2012-1174, CVE-2012-0871
- Bugs médiatiques : CVE-2017-1000082, CVE-2017-9445, CVE-2017-15908
- Ecrit en C mais la majeure parties des bugs sont logiques (non liées au langage)
- Surface d'attaque assez limitée
- Démons systèmes correctement confinés (voir les vulnérabilités médiatiques sur `resolved`)

Suite

Capabilities, seccomp-bpf, NoNewPrivileges